CLAIMS

What is claimed is:

1. A method in a computer system of executing a first sequence of modules in a first task, said first sequence of modules linked to one another and having at least one sequence of execution, comprising the following steps:
   a. storing in each of said first sequence of modules a skip value indicating a next module in said sequence of modules to execute;
   b. executing a first module of said first sequence of said modules; and
   c. executing said next module of said sequence of modules indicated by the skip value.

2. The method of claim 1 wherein the skip value comprises the integer N which indicates that execution should skip to the N+1th module following execution of a currently executed module in the first sequence of modules.

3. The method of claim 2 wherein a value of N less than zero associated with the currently executed module indicates that execution of the first sequence of modules should terminate after completion of execution of the currently executed module.

4. The method of claim 1 further comprising executing steps a-c on a second sequence of modules, said second sequence of modules becoming the first sequence of modules, said second sequence of modules being a part of a second task.

5. The method of claim 4 further comprising repeating steps a-c on the first sequence of said modules, said first task and the second task being linked with one another in a task list.

6. The method of claim 1 further comprising performing the skip action created on the previous iteration of the first set of modules.

7. The method of claim 1 wherein the skip value associated with each module in said first sequence of modules may be modified by a module associated with the skip value.

8. The method of claim 1 wherein the skip value associated with each module in said first sequence of modules may be modified by a host associated with the first task.

9. A method of controlling execution flow of a first task comprising a sequence of first executable modules in a processing system by associating with each of said first executable modules a skip count, said skip count comprising an integer N which indicates that execution should skip to the N+1th module following execution of a currently executed module in the first sequence of executable modules, a value of N less than zero associated with the currently executed module indicating that execution of the first sequence of modules should terminate after completion of execution of the currently executed module.

10. A method performed by a processor of controlling the flow of execution of a first set of executable modules sequentially associated with one another comprising the following steps:

a. executing a first module in said first sequence of modules;

b. determining a skip value associated with said first module; and

c. proceeding to execute a subsequent module in said first set of executable modules indicated by said skip value.

11. The method of claim 10 wherein the skip value comprises the integer N, and the subsequent module is the N+1th module following the first module in said first sequence of modules.

12. The method of claim 11 wherein a value of N less than zero associated with said first module indicates that execution of said first sequence of modules should terminate after completion of execution of said first module.

13. The method of claim 10 wherein the skip value of each module of said first sequence of modules is stored in a datum associated with said each module.

14. The method of claim 10 wherein the said skip value in each of said modules may be modified by a host.

15. The method of claim 10 wherein the said skip value in each of said modules may be modified by each of the respective modules in said first sequence of modules.

16. An apparatus for executing a first sequence of modules in a first task, said first sequence of modules linked to one another and having at least one sequence of execution, comprising:

a. means for storing in each of said first sequence of modules a skip value indicating a next module in said sequence of modules to execute;

b. means for executing a first module of said first sequence of said modules; and

c. means for executing said next module of said sequence of modules indicated by the skip value.

17. An apparatus for controlling the flow of execution of a first set of executable modules sequentially associated with one another comprising:

a. means for executing a first module in said first sequence of modules;

b. means for determining a skip value associated with said first module; and

c. means for proceeding to execute a subsequent module in said first set of executable modules indicated by said skip value.

18. A method of controlling the execution sequence of a series of modules by a processor, each of said modules associated with one another, comprising the following steps:

a. executing the first in said series of modules;

b. determining a skip value N associated with said first in said series of said modules;

c. if the skip value N associated with said first module is less than zero, then terminating the execution of said series of modules;

d. else if the skip value N associated with said first module is greater than or equal to zero then proceeding to a N+1th module in said series of said modules.

19. A method in a computer system of performing a first sequence of modules in a first task, said first sequence of modules linked to one another and having at least one sequence of execution, comprising the following steps:

a. storing in a first module of said first sequence of modules a skip value N representing a subsequent module in said first sequence of modules to execute, said skip value N comprising either:

i. an integer less than zero indicating that said first module is a last executable module to be executed in said sequence of modules;

ii. an integer greater than or equal to zero indicating that said process should proceed to said N+1th module subsequent to said first module in said first sequence of said modules;

b. executing the first of said first sequence of said modules; and

c. executing the subsequent module in said sequence of said modules indicated by said skip value.

20. A method of controlling the activation of a sequence of tasks by a processing system comprising at least one processor, comprising the following steps:

a. determining tasks which require synchronization;

b. adding a reference to each of said tasks to a synchronization list said reference including a request to activate/deactivate the task and a frame offset in which to activate/deactivate the task;

c. for each task in said synchronization list, flagging said task for activation/deactivation and a frame in which to activate/deactivate the task based on said frame offset;

d. then, for each time frame subsequent to the completion of the performance of step c, performing the following steps for each task in said processing system:

i. determining whether a requested task status is equal to a current status of said task;

ii. if the requested task status is not equal to the current status and a number of subsequent frames executed equals the frame offset then modifying the state of the task to the requested state; and

iii. if the state of the task is active then executing said task.

21. The method of claim 20 wherein steps a-c are performed in a host processor and step d is performed in a slave processor.

22. The method of claim 21 wherein said slave processor is a digital signal processor (DSP).

23. The method of claim 20 wherein the said tasks in said processing system are stored in a task list.

24. The method of claim 20 wherein the processing system comprises a frame-based processing system.

25. The method of claim 20 further comprising disposing of said synchronization list upon the synchronization of each task in said synchronization list.

26. The method of claim 20 further comprising disabling interrupts in said processing system after the creation of said synchronization list and before beginning the synchronization of each task in said synchronization list, and enabling interrupts upon the completion of synchronization of said tasks.

27. A method of controlling the activation of a sequence of tasks by a frame-based processing system comprising at least one processor, comprising the following steps:

a. determining each task which requires synchronization in said processing system, flagging a requested task status for said task and a frame in which to modify the task status;

b. then, for each time frame subsequent to the completion of the performance of step a, performing the following steps for each task in said processing system:

i. determining whether said requested task status is equal to a current status of said task;

ii. if the requested task status is not equal to the current status and a number of subsequent frames executed

equals the frame offset then modifying the state of the task to the requested state; and

iii. if the status of the task is active then executing said task.

28. The method of claim 27 wherein said processing system comprises a host processor and a slave processor.

29. The method of claim 27 wherein the said tasks in said processing system are stored in a task list.

30. An apparatus for controlling the activation of a sequence of tasks by a processing system comprising at least one processor, comprising:

a. means for determining tasks which require synchronization;

b. means for adding a reference to each of said tasks to a synchronization list said reference including a request to activate/deactivate the task and a frame offset in which to activate/deactivate the task;

c. means for flagging said tasks in said synchronization list for activation/deactivation and a frame in which to activate/deactivate the task based on said frame offset;

d. means for activating the following apparatus for each task in said processing system in each time frame subsequent to the activation of the apparatus in element c comprising:

i. means for determining whether a requested task status is equal to a current status of said task;

ii. means for modifying the state of the task to the requested state if the requested task status is not equal to the current status and a number of subsequent frames executed equals the frame offset; and

iii. means for executing said task if the state of the task is active.

31. The apparatus of claim 30 wherein the means of elements a-c comprise a host processor and element d comprises a slave processor.

32. The apparatus of claim 31 wherein said slave processor is a digital signal processor (DSP).

33. The apparatus of claim 30 further comprising a means for storing references to said tasks in said processing system in a task list.

34. The apparatus of claim 30 further comprising a frame-based apparatus.

35. The apparatus of claim 30 further comprising means for disposing of said synchronization list upon the synchronization of each task in said synchronization list.

36. The apparatus of claim 30 further comprising means for disabling interrupts in said processing system after the creation of said synchronization list and prior to beginning the synchronization of each task in said synchronization list, and means for enabling interrupts upon the completion of synchronization of said tasks.

37. An apparatus for controlling the activation of a sequence of tasks by a processing system comprising at least one processor, comprising:

a. means for determining each of said tasks which require synchronization, a request to activate/deactivate the task and a frame offset in which to activate/deactivate the task;

b. means for activating the following apparatus for each task in said processing system in each time frame subsequent to the activation of the apparatus in element a comprising:

i. means for determining whether a requested task status is equal to a current status of said task;

ii. means for modifying the state of the task to the requested state if the requested task status is not equal to the current status and a number of subsequent frames executed equals the frame offset; and

iii. means for executing said task if the state of the task is active.

38. The apparatus of claim 37 wherein said processing system comprises a host processor and a slave processor.

39. The apparatus of claim 37 further comprising a means for storing references to said tasks in said processing system in a task list.

40. A method of controlling the activation of a sequence of tasks by a processor, each of said tasks normally executed in a sequential fashion by said processor, comprising the following steps:

a. determining the state of a simultaneous execution flag;

b. if said simultaneous execution flag is not set, then executing a first task and terminating;

c. if said simultaneous execution flag is set, then determining if the client which references the first task has control of the semaphore, and if not, then terminating;

d. if said simultaneous execution flag is set and the client which references the first task has control of the semaphore then determining if a toggle active flag condition is present;

e. if the toggle active flag is set then toggling the state of the first task execution flag and terminating;

f. if the toggle active flag is not set then determining the state of the first task execution flag; and

g. if the first task execution flag is set then executing the first task, otherwise halting the first task.

41. The method of claim 40 further comprising performing steps a-g for a second task in the sequence of tasks, said second task coming after the first task in the sequence of tasks, said second task becoming the first task.

42. The method of claim 40 further comprising repeating steps a-g for each task in said sequence of tasks, each task becoming the first task.

43. The method of claim 42 further comprising repeating steps a-g until all tasks in said sequence of tasks has been serviced, and then, performing a-g on said first task.

44. The method of claim 40 wherein said sequence of tasks comprises a series of real-time tasks.

45. The method of claim 40 wherein said processor comprises a digital signal processor (DSP).

46. The method of claim 40 wherein said first task active flag is contained in a datum which is associated with said first task.

47. The method of claim 40 wherein said first task is comprised of a series of modules associated with one another.

48. The method claim of 47 wherein each of said modules is linked in order of execution flow.

49. The method of claim 43 wherein performing steps a-g on each of said tasks is performed within a signal frame of said processor's execution.

50. An apparatus for controlling the activation of a sequence of tasks by a processor, each of said tasks normally executed in a sequential fashion by said processor comprising:

a. means for determining the state of a simultaneous execution flag;

b. means for executing a first task and terminating if said simultaneous execution flag is not set;

c. means for determining if the client which references the first task has control of the semaphore if said simultaneous execution flag is set;

d. means for halting execution of the first task if the client which references the first task does not have control of the semaphore;

e. means for determining the state of a toggle active flag if said simultaneous execution flag is set and the client which references the first task has control of the semaphore;

f. means for toggling the state of the first task execution flag and terminating if the toggle active flag is set;

g. means for determining the state of the first task execution flag if the toggle active flag is not set;

h. means for executing the first task if the first task execution flag is set; and

i. means for halting the first task if the first task execution flag is not set.

51. The apparatus of claim 50 wherein said apparatus further comprises a means for applying said apparatus to each of the tasks in a task list in sequence.

52. The apparatus of claim 51 wherein said last task in said task list comprises a reference to a first task in said task list.

53. The apparatus of claim 50 wherein said task list comprises real-time tasks.

54. The apparatus of claim 50 wherein said processor comprises a digital signal processor (DSP).

55. The apparatus of claim 50 further comprising a means for activating each of said means in said apparatus for each task in said task list is performed within a frame of said processor's execution.

B